М.С. Брагин, О.В.Ниссенбаум
Тюменский государственный университет, г.Тюмень

ПУТИ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОТ ВЗЛОМА

Обмен информацией стал важной составляющей современного общества. Мы полагаемся на встроенное программное обеспечение в наших автомобилях, в наших домах, каждый день используем электронные устройства. Использование программного обеспечения стало обыденным процессом в нашей повседневной жизни. Очевидно, что приложения полагаются на правильное функционирование программного и аппаратного обеспечения. В эпоху программного обеспечения доходы компаний-разработчиков огромны. Не только операционные системы, но также и профессиональные приложения (например, графические редакторы) могут быть очень дорогим. Как следствие, такое ПО взламывается и незаконно распространяется. Всего за несколько щелчков мыши можно загрузить программное обеспечение, применить загруженный патч, и начать использовать его без оплаты. Тем не менее, разработчики противостоят таким угрозам. За последние 30 лет с начала развития защиты программ от копирования появились различные техники, подходы к решению данной задачи.[1]

Рассмотрим сферу коммерческого ПО, ведь именно такое ПО в первую очередь требует защиты. Обычно, такие программы нужно защищать от:

1) Несанкционированного копирования;

2) Восстановления алгоритмов, извлечения ключей шифрования, подписи;

3) Изменения алгоритмов программ, подделки.

В первом случае злоумышленник просто отключает проверку лицензирования ПО, и распространяет “исправленную” программу, зачастую не бесплатно. Во втором случае проводится более тщательный анализ скомпилированного кода, цели могут преследоваться различные, от разработки аналога до компрометации информации легитимных пользователей ПО. В

третьем случае возможно исправление ошибок разработчика, например, после окончания поддержки продукта, или выдача взломщиком своего собственного ПО в качестве настоящего. Все эти случаи объединяет тот момент, что злоумышленник обладает всем скомпилированным кодом программы и может прибегнуть к инструментам реверс-инжиниринга.

С конца прошлого века большую популярность приобрели высокоуровневые языки программирования, компилирующиеся в промежуточный байт-код для исполнения на виртуальной машине(ВМ)[2]. Программы написанные с использованием таких ЯП обладают особенностью, восстановить исходный код, разобраться в алгоритме гораздо проще, чем для компилирующихся в машинные коды ЯП. Причиной тому служит меньший набор команд ВМ, их простота в сравнении с инструкциями процессора. Команды виртуальной машины доступны для анализа даже неискушенному взломщику. Поэтому для интерпретируемых языков баланс между сложностью защиты и взлома ПО сдвигается в сторону упрощения взлома. Отсюда возникает необходимость в эффективных способах защиты.

Широко используемые виды защит, такие как обфускация, упаковка, изменение hex значений инструкций ВМ усложняют анализ, но полностью проблему не решают. Программный код остается доступным для анализа и отладки. Стоит так же отметить, что виды защит, основанные на предположении, что проверку условия принято/неПринято(будь то регистрационный ключ, проверка сертификата, целостности программного кода) нельзя отключить в общем случае несостоятельны, ведь в программный код можно вмешаться.

Перспективными выглядят пути: выполнение кода на стороне сервера, вплоть до полного переноса ПО в online в совокупности с тонким клиентом; использование аппаратных ключей с программируемым алгоритмом. В обоих случаях стойкость защиты основана на том, что ключевая информация(исполняемый код, криптографические ключи) не покидают устройство защиты в процессе работы с ним. Первый подход зависит в первую

очередь от стойкости самих серверов, так же становится важно обеспечение доступности сервиса, поддержания необходимой инфраструктуры, что может свести на нет все преимущества данного подхода. Кроме того, не всегда, особенно в коммерческой сфере, есть доступ к сети Интернет.

Учитывая все вышесказанное, привлекательно выглядит следующее решение – использование внешнего аппаратного ключа защиты. Такой ключ представляет собой платформу для выполнения кода с интерфейсом для обмена данными с ПК (usb/Ethernet). Разработчику при написании программы нужно выбрать участки кода(функции), которые будут выполняться на внешнем ключе. Эти участки кода являются критичными для функционирования программы (например в Microsoft Office Word функционал открытия/сохранения файлов, без которых он практически бесполезен), либо критичными для защиты(ноу-хау производителя, например способ сжатия двоичных последовательностей архиватором winRar). Далее эти функции вызываются через API ключа, аналогично функциям подключаемых библиотек.

Для пользователя приобретение программного продукта состоит в приобретении аппаратного ключа. При использовании программы требуется подключить ключ к ПК.

СПИСОК ЛИТЕРАТУРЫ

1. Nate Lawson. Software protection introduction // Rdist [Электронный ресурс] URL: https://rdist.root.org/2007/03/21/software-protection-introduction/ (дата обращения 20.04.2017г).

2. Stephen O'Grady. The RedMonk Programming Language Rankings: January 2017 // RedMonk. [Электронный ресурс] URL: http://redmonk.com/sogrady/2017/03/17/language-rankings-1-17/ (дата обращения 21.04.2017г).